cin.ufpe.br

Centro de Informática

U · F · P · E

Universidade Federal de Pernambuco

CPU: Estrutura e Funcionalidade

Roteiro da Aula

- Ciclo de Instrução
- Projeto de uma CPU simples: conceitos
- Componentes básicos
  - Leitura da instrução
  - Operação entre registradores
  - Acesso à memória
- Implementação Mono-ciclo
  - Leitura da Instrução
  - Operações Aritméticas
  - Leitura + Operação entre registradores
  - Acesso à Memória
  - Desvio Condicional
- Visualizando a execução da instrução
- Análise de Desempenho

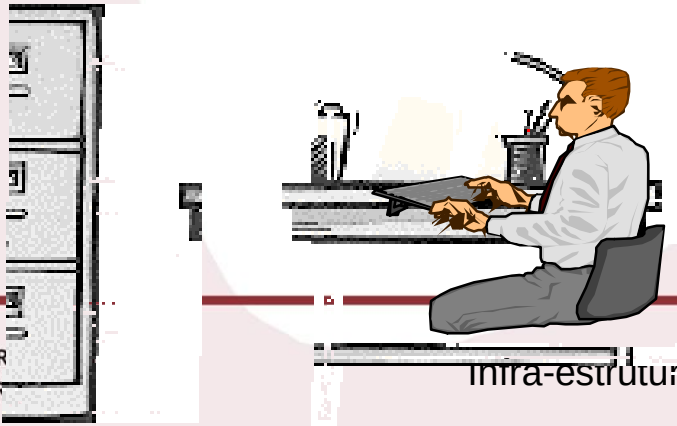
Como um documento é processado?

1- Busca documento

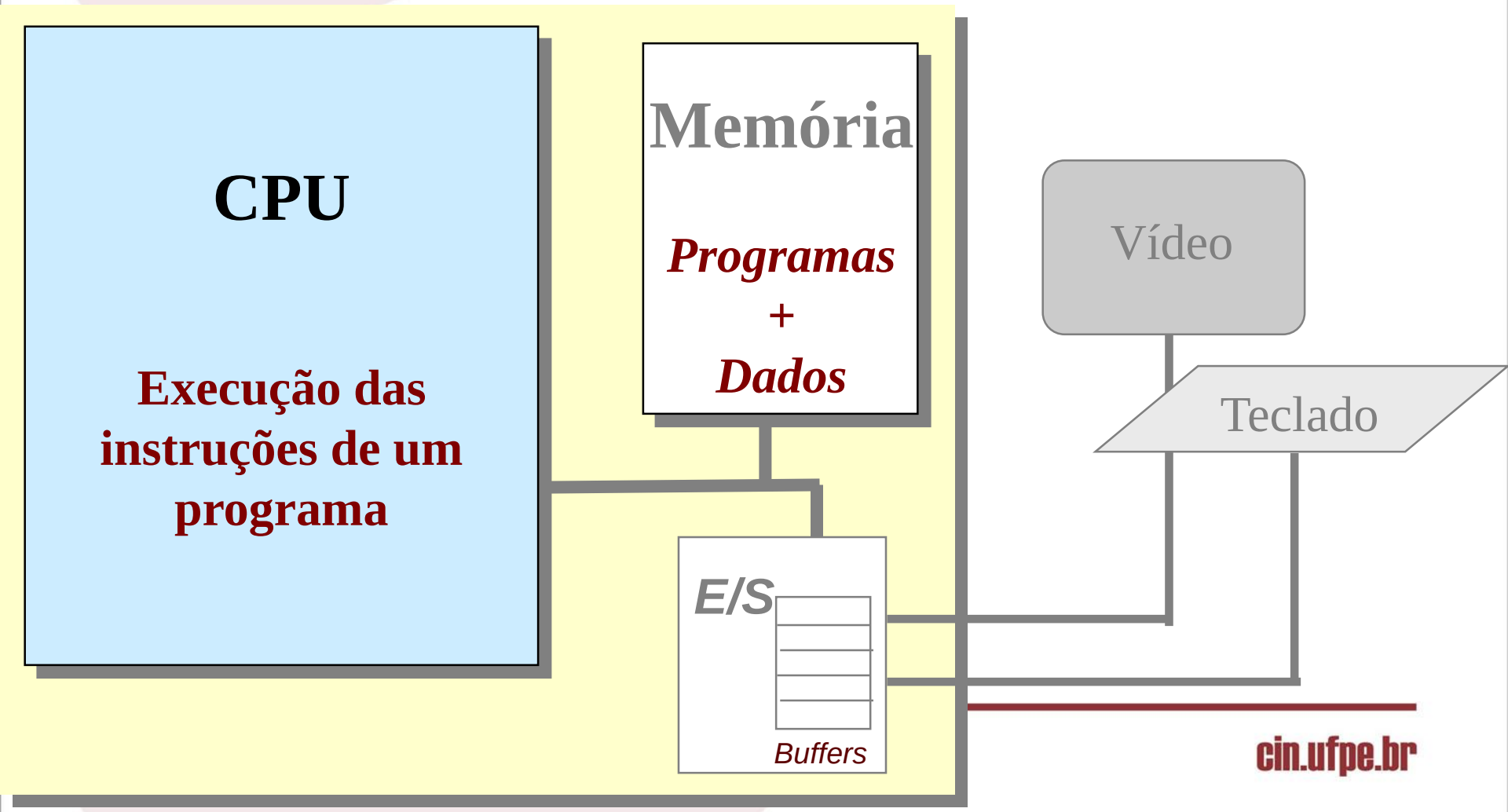
2 - Identifica tipo de transação

3 - Verifica se saldo é positivo

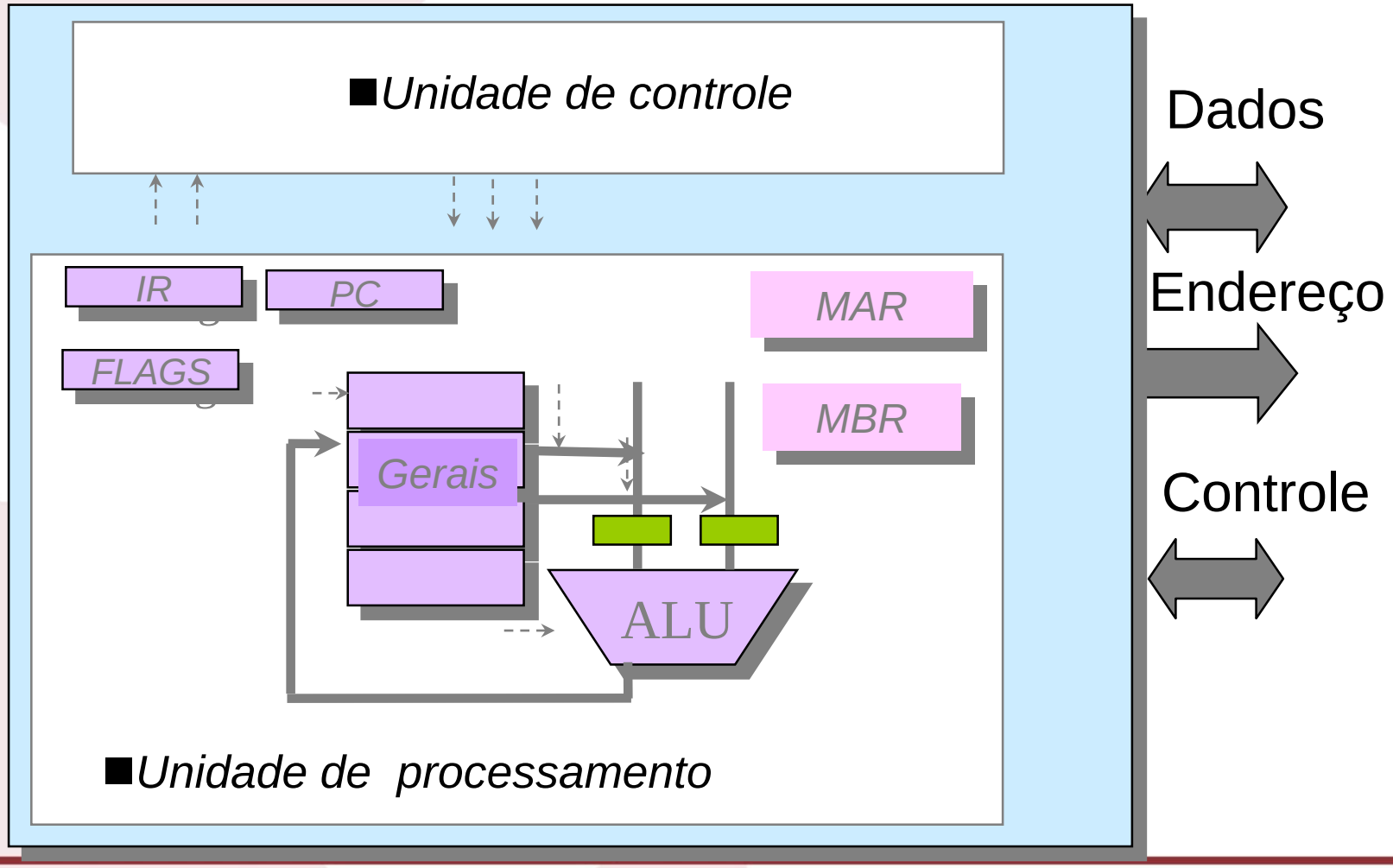
4 - Efetiva transação e atualiza saldo



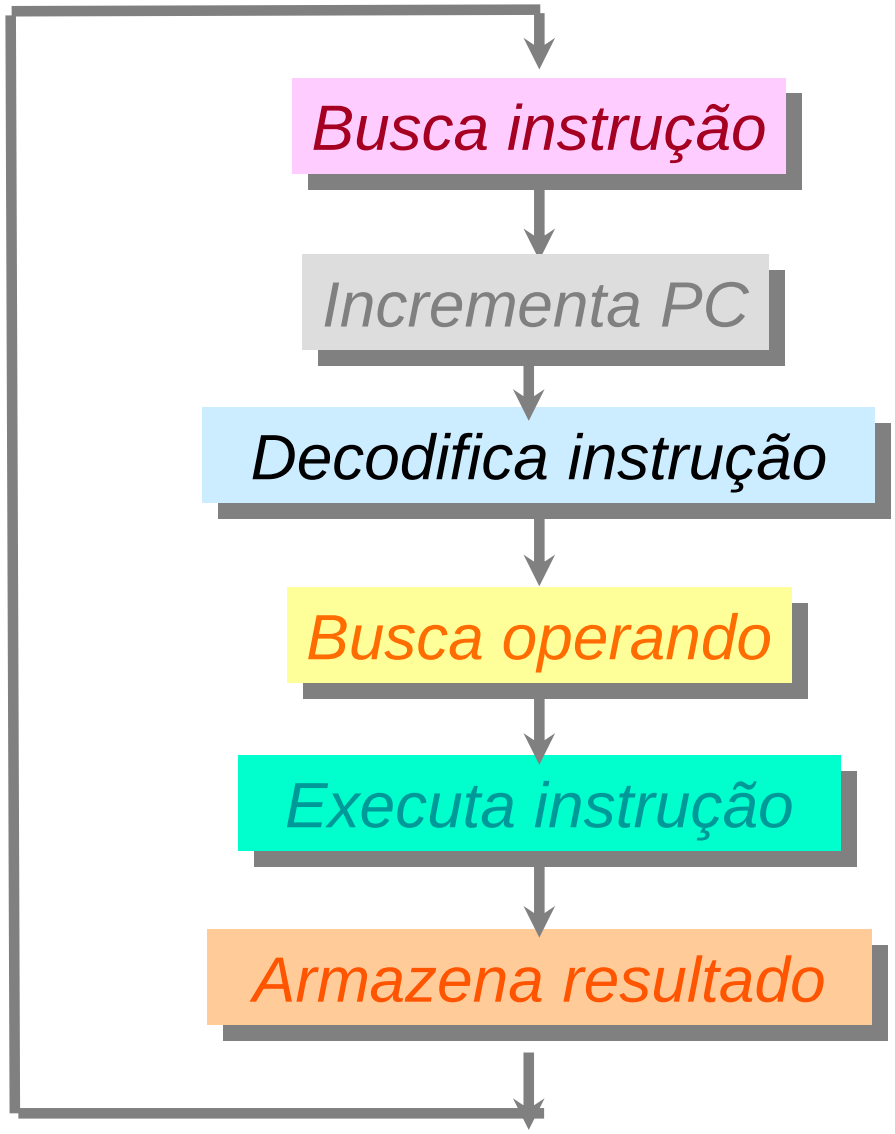
Componentes de um computador



Unidade Central de Processamento



Ciclo de Instrução



Projeto de uma Arquitetura

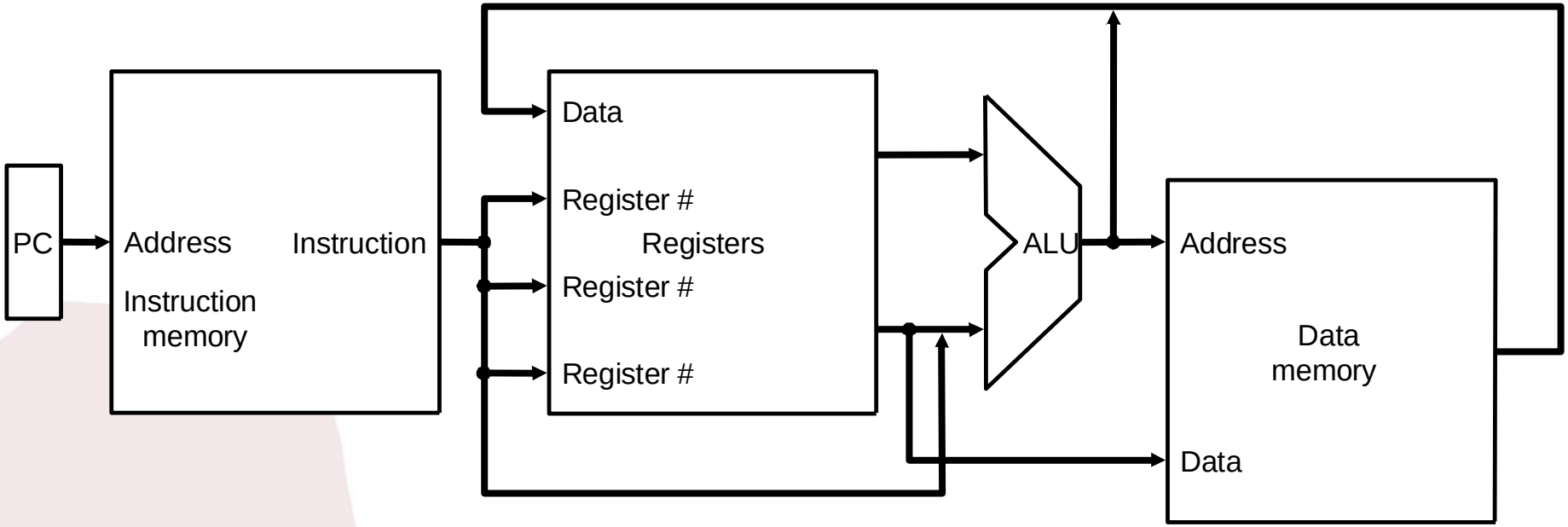
- Conjunto de registradores
- Tipos de Dados
- Formato e Repertório de instruções

Projeto: uma CPU simples...

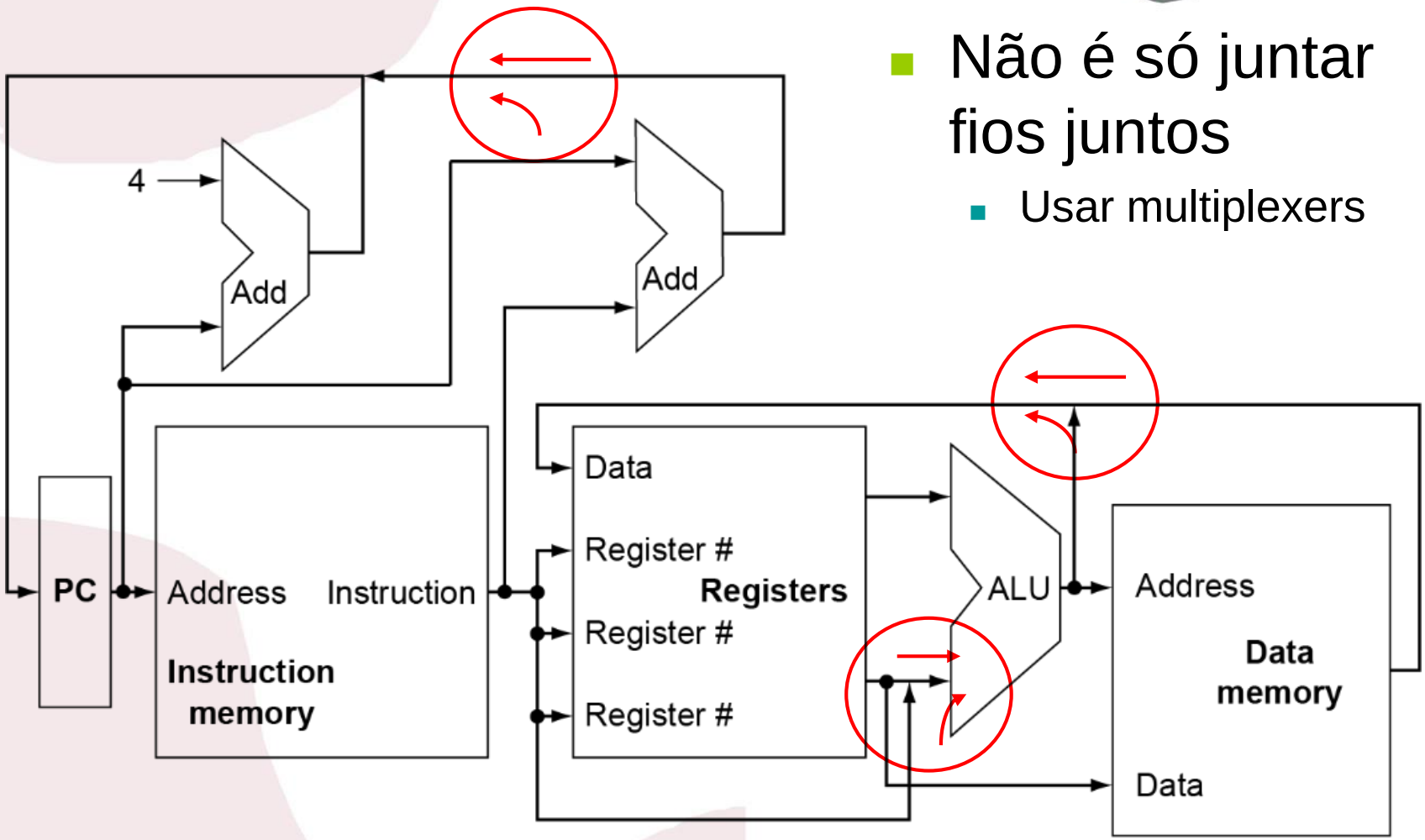
Instrução	Descrição
LD rt, desl(rs)	Carrega palavra de mem em rs
SD rt, desl(rs)	Armaz. Reg. na memória
ADD rd, rs, rt	rd <- rs + rt
SUB rd, rs, rt	rd <- rs – rt
AND rd, rs, rt	rd <- rs and rt
BEQ rs, rt, end	Desvio se rs = rt

Name (Field Size)	Field 7 bits	5 bits	5 bits	3 bits	5 bits	7 bits	Comments
R-type	funct7	rs2	rs1	funct3	rd	opcode	Arithmetic instruction format
I-type	immediate[11:0]		rs1	funct3	rd	opcode	Loads & immediate arithmetic
S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode	Stores
SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode	Conditional branch format
UJ-type	immediate[20,10:1,11,19:12]				rd	opcode	Unconditional jump format
U-type	immediate[31:12]				rd	opcode	Upper immediate format

RISC V - Visão Abstrata

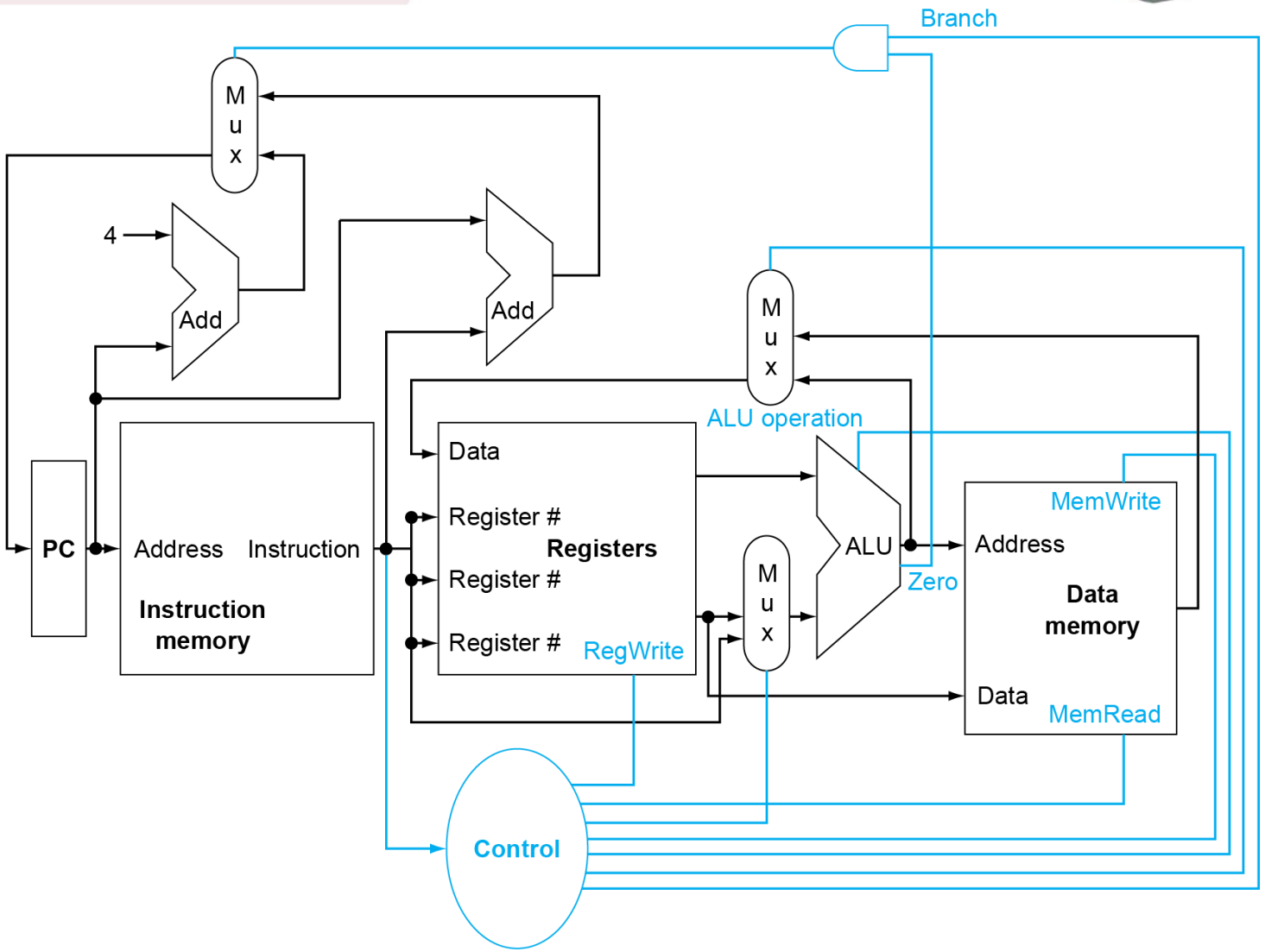


Multiplexers



- Não é só juntar fios juntos
  - Usar multiplexers

Controle

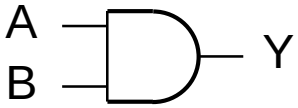

Branch
M
u
x
4
Add
Add
M
u
x
ALU operation
PC
Address
Instruction
Instruction
memory
Data
Register #
Registers
Register #
Register #
RegWrite
M
u
x
ALU
Zero
MemWrite
Address
Data
memory
Data
MemRead
Control

Conceitos Básico Projeto Lógico

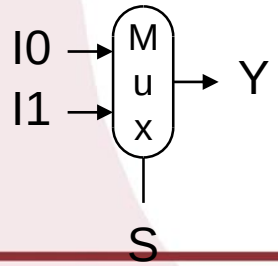
- Informação codificada em binário
  - Baixa tensão = 0, alta tensão = 1
  - Um fio por bit
  - Dados codificados como vários bits são implementados como barramentos
- Circuito combinacional
  - Operam os dados
  - A saída é uma função da entrada
- Circuitos de estado (sequenciais)
  - Guardam informação

Elementos Combinacionais

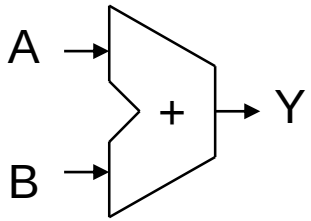
- AND-gate
  - Y = A & B



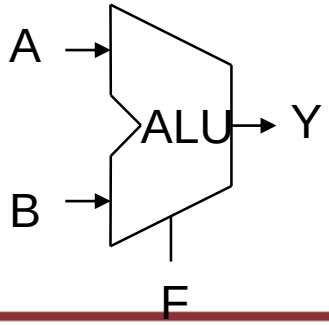
- Multiplexer
  - Y = S ? I1 : I0



- Adder
  - Y = A + B

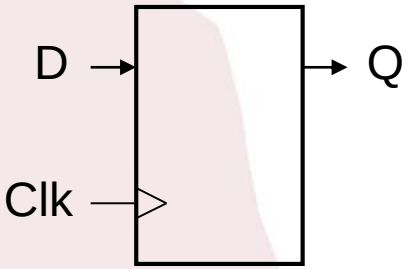


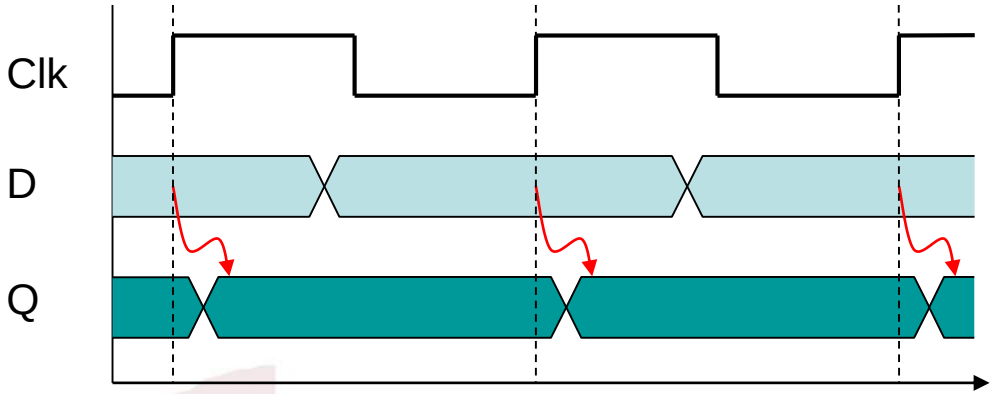
- Arithmetic/Logic Unit
  - Y = F(A, B)



Elementos Sequenciais

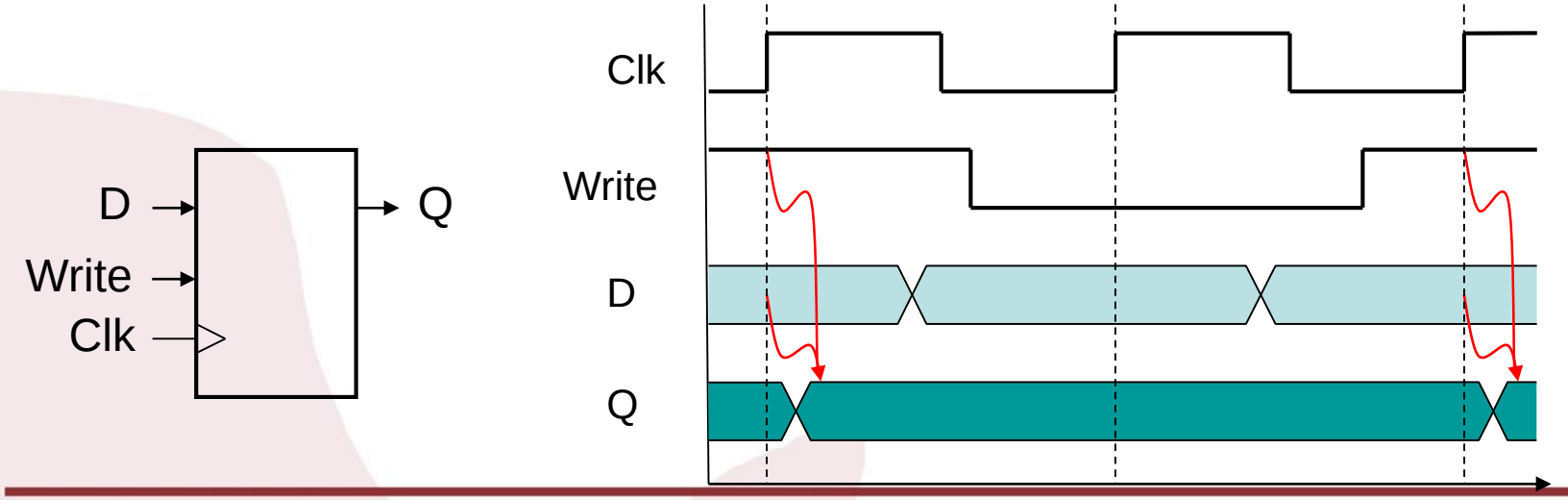
- Registrador: armazena dados em um circuito
- Usa um sinal do clock para determinar quando atualizar o valor armazenado
  - Edge-triggered: atualiza quando Clk muda de 0 para 1





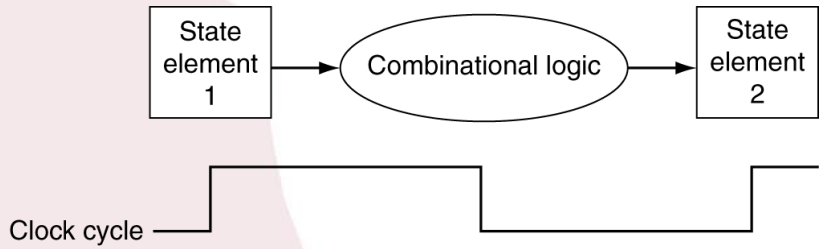
Elementos Sequenciais

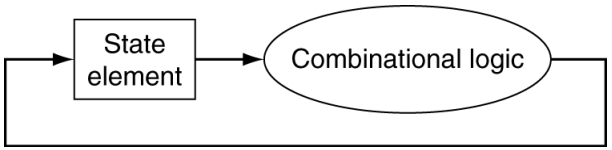
- Registrador com controle de escrita
  - Sómente atualiza na transição do clock edge quando controle write é 1
  - Usado quando valor armazenado será usado posteriormente



Uso do Clock

- A lógica combinacional transforma os dados durante os ciclos de clock
- Entre transições do clock
  - Entrada de elementos de estado, saída para o elemento de estado
  - Atraso mais longo determina o período do relógio

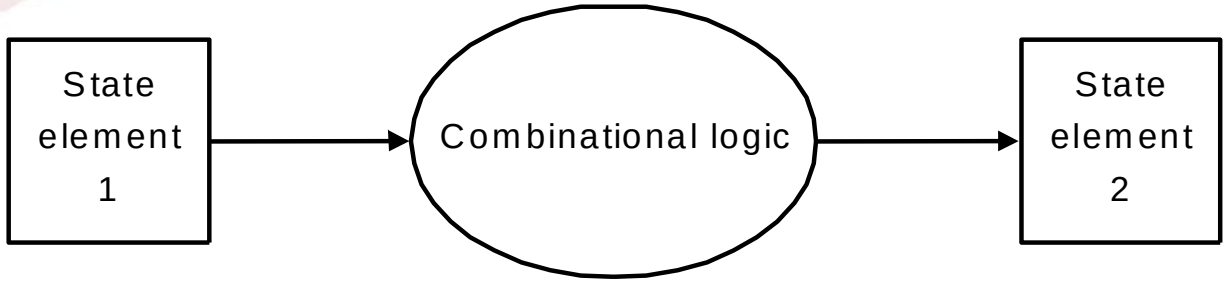


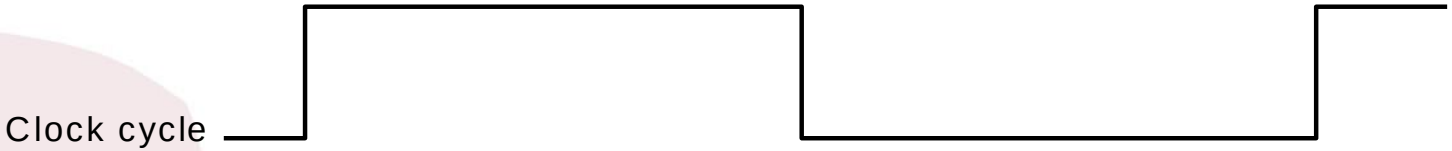


Projetando um Datapath

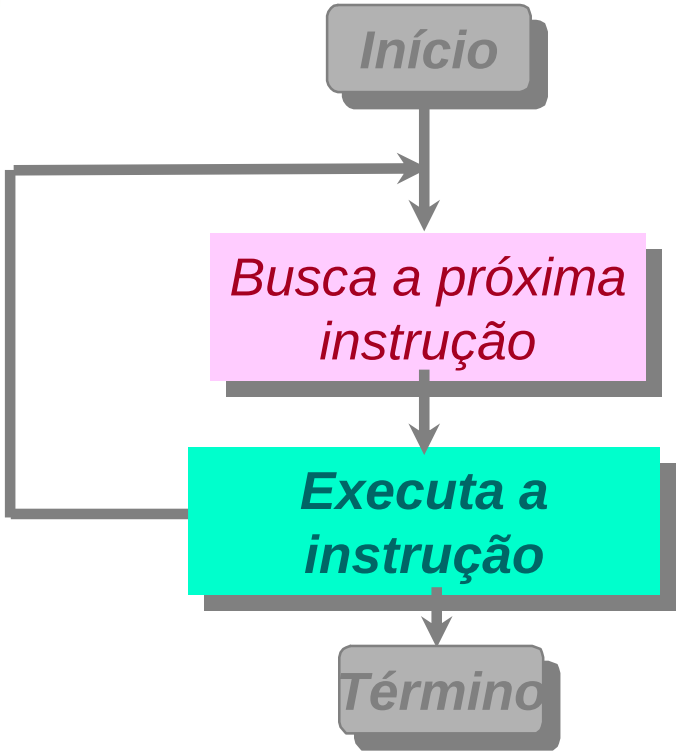
- Datapath (Caminho de Dados)
  - Elementos que processam dados e endereços na CPU
    - Registradores, ALUs, mux's, memórias,…
- Vamos construir um caminho de dados RISC-V incrementalmente
  - Refinando o projeto da versão geral

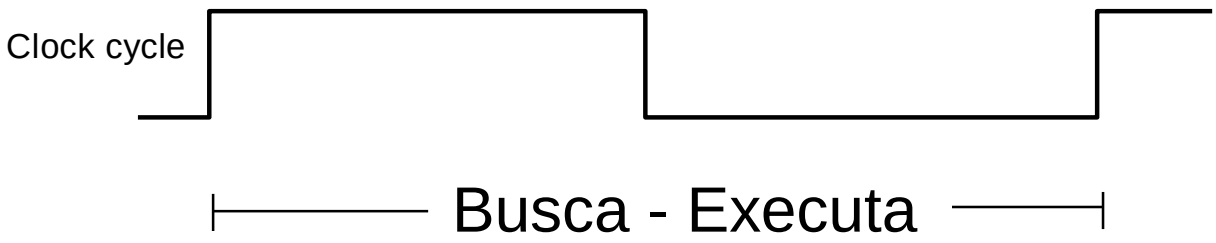
Relógio - Clock



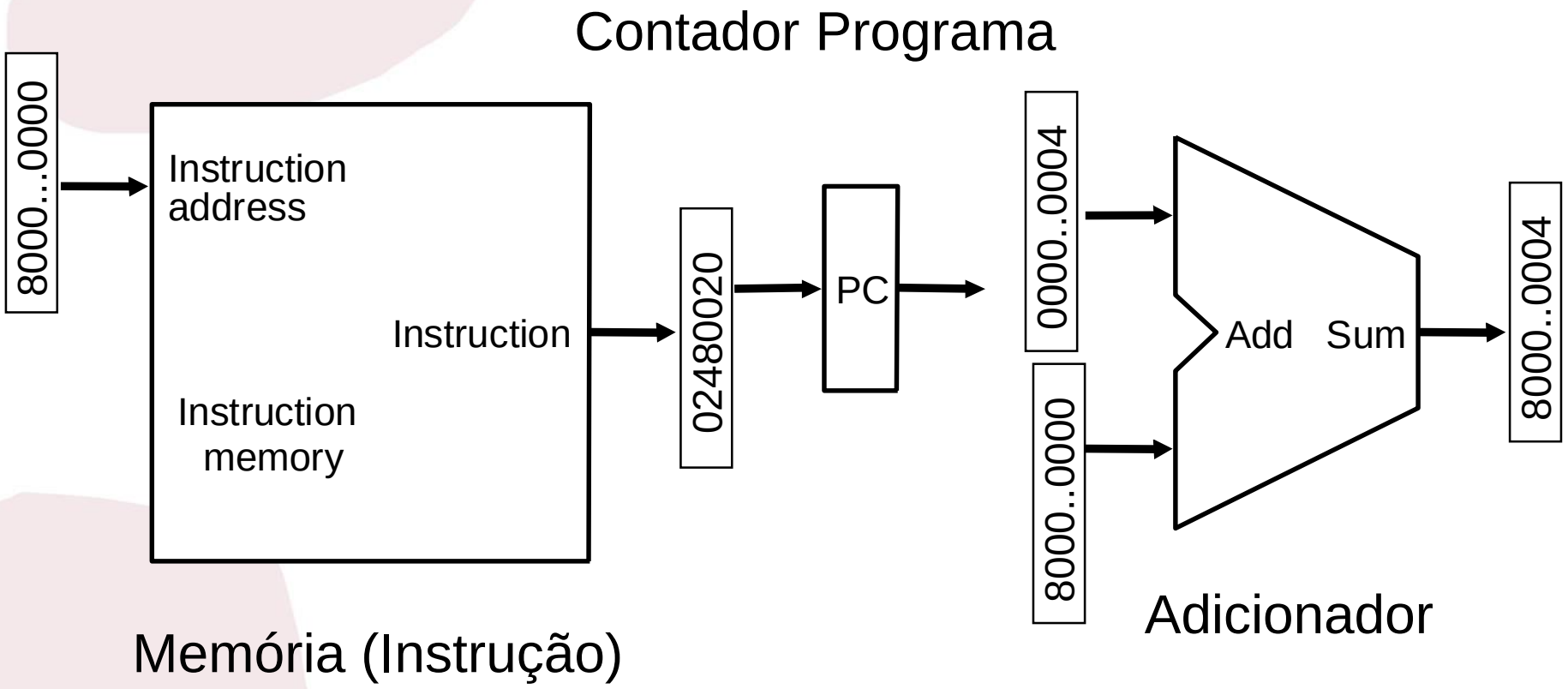


Mono-ciclo

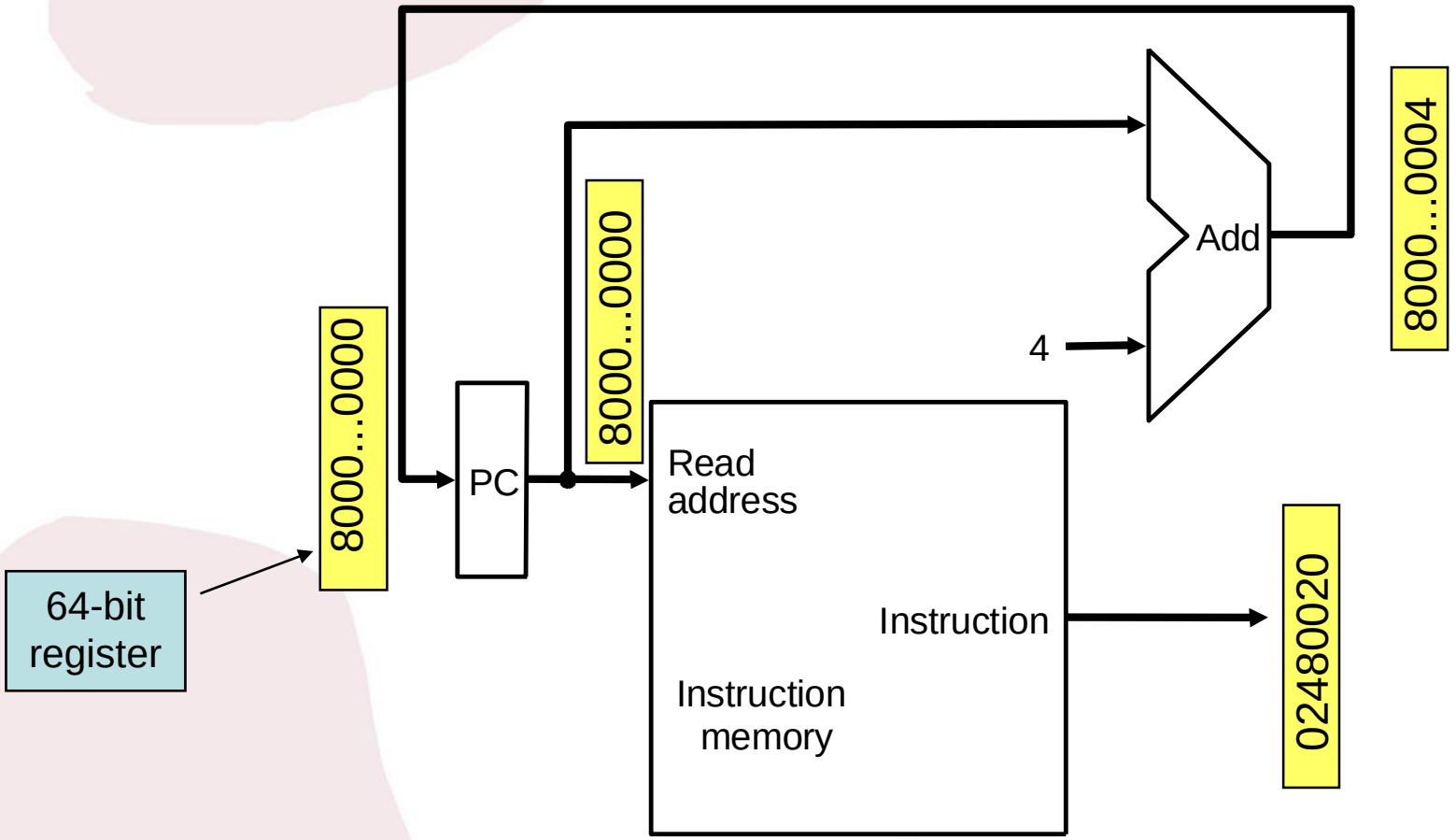

Início
Busca a próxima instrução
Executa a instrução
Término


Clock cycle
Busca - Executa

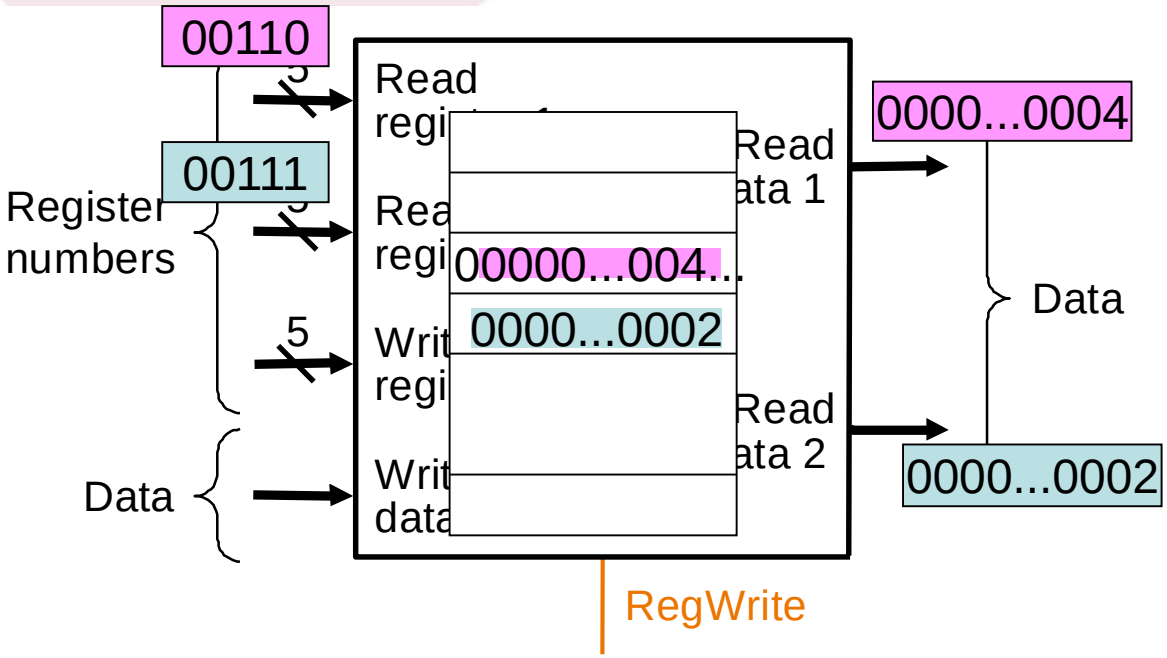
Componentes Básicos: Busca de Instrução



Busca de Instrução

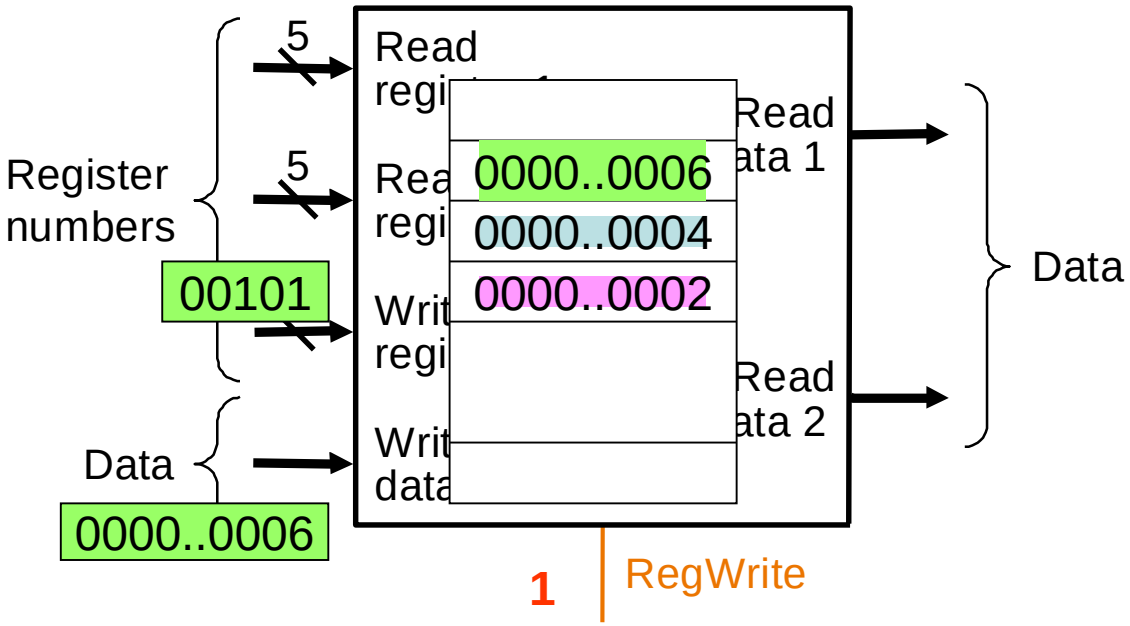


Componentes Básicos: Operações Aritméticas



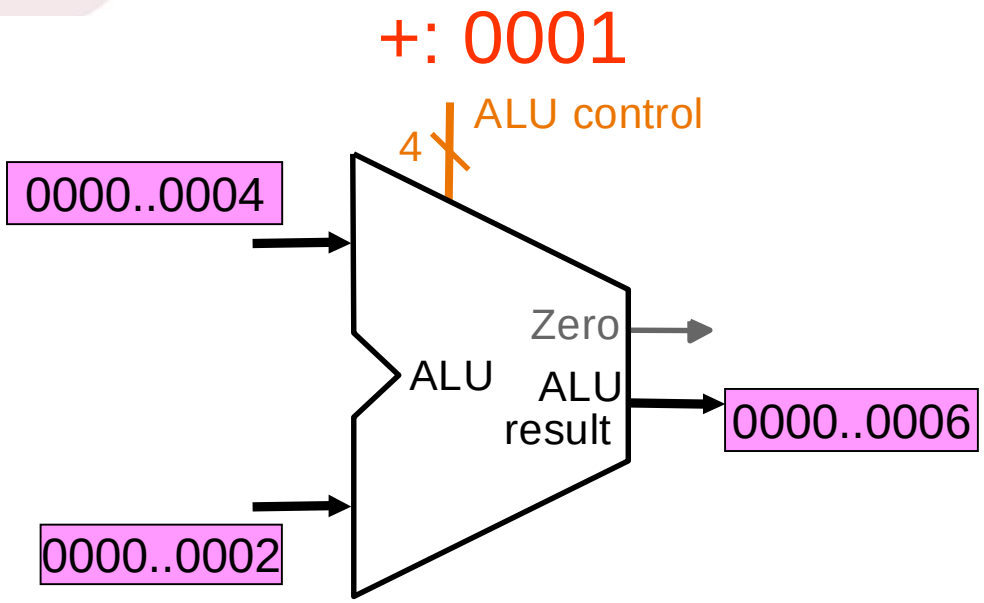
Banco de registradores
Leitura

Componentes Básicos: Operações Aritméticas



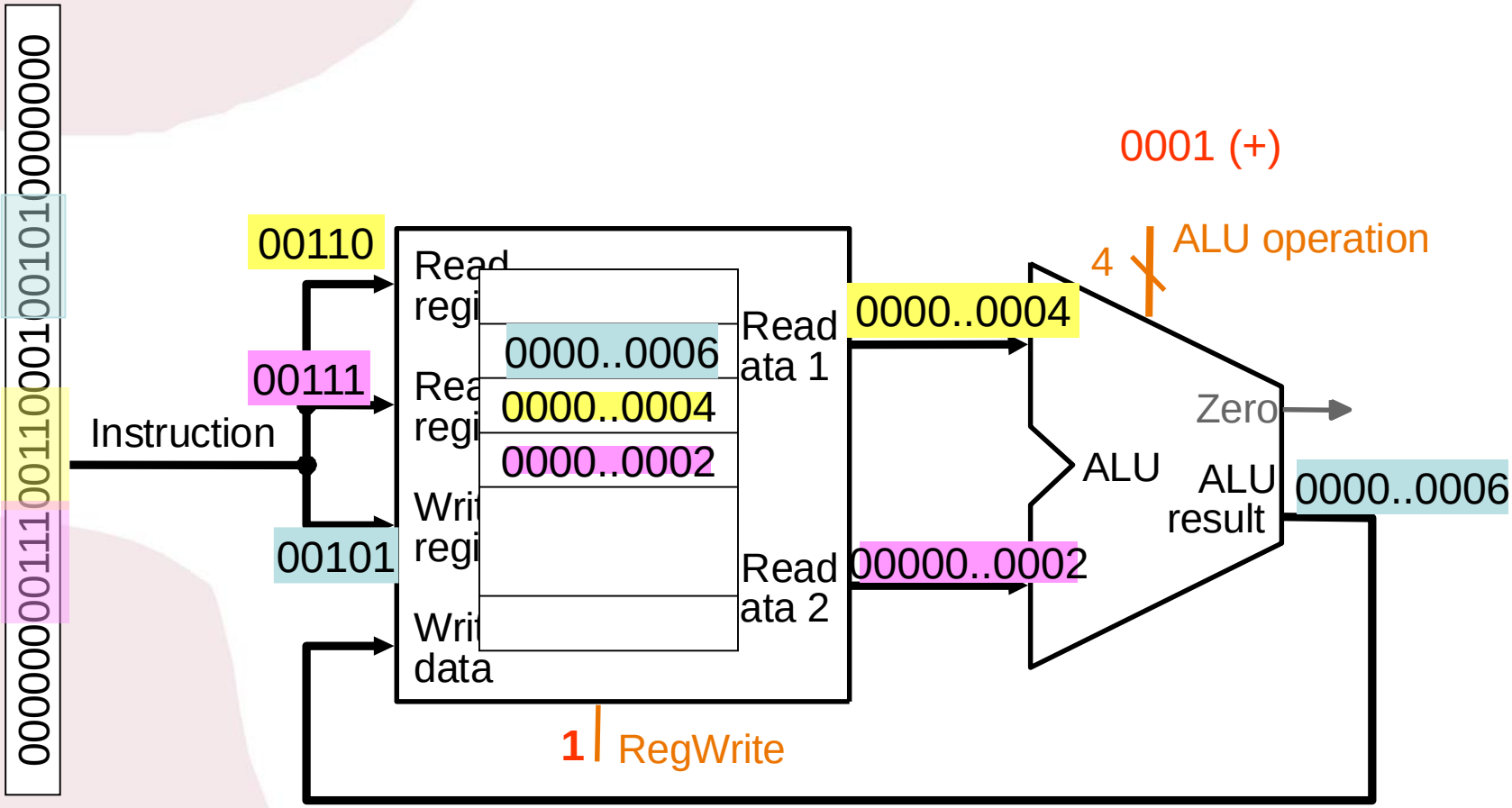
Banco de registradores
Escrita

Componentes Básicos: Operações Aritméticas

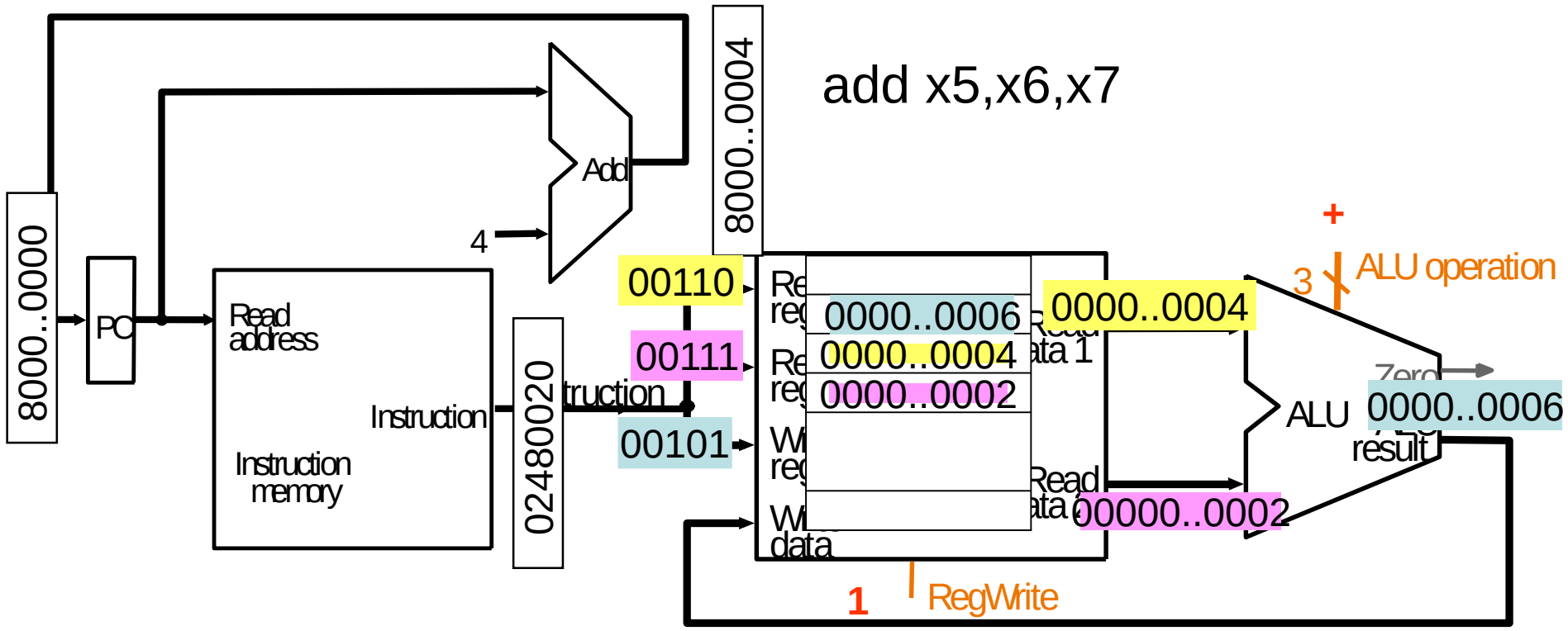


ALU

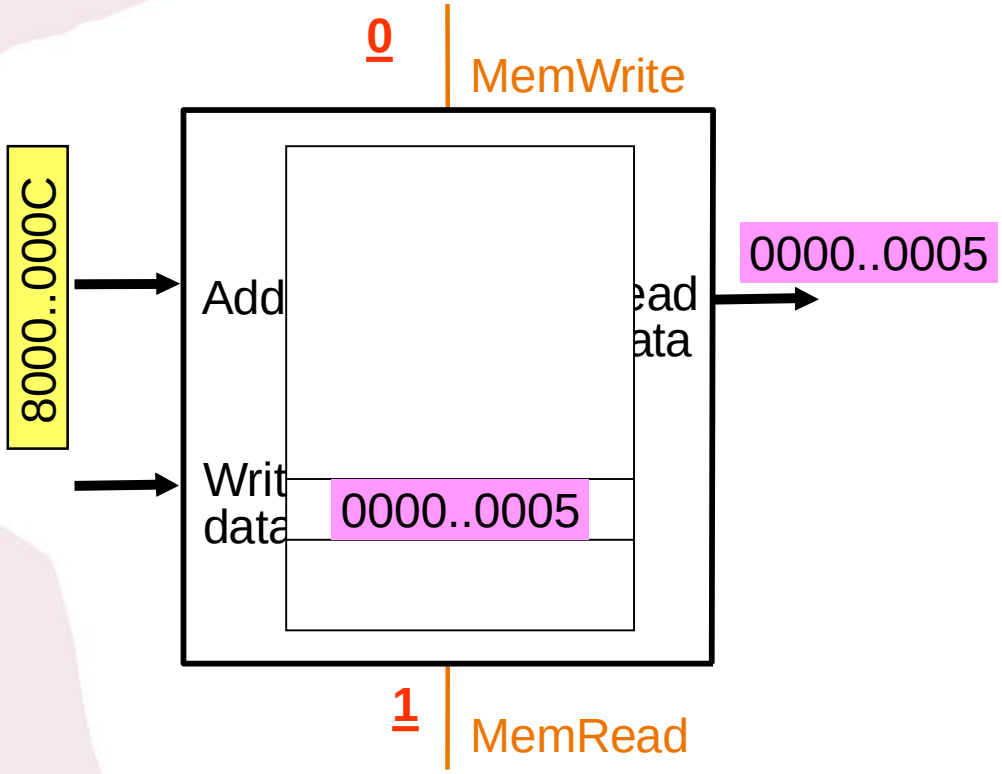
Instruções Aritméticas/Lógicas



Busca e Execução de Instruções Aritméticas/Lógicas

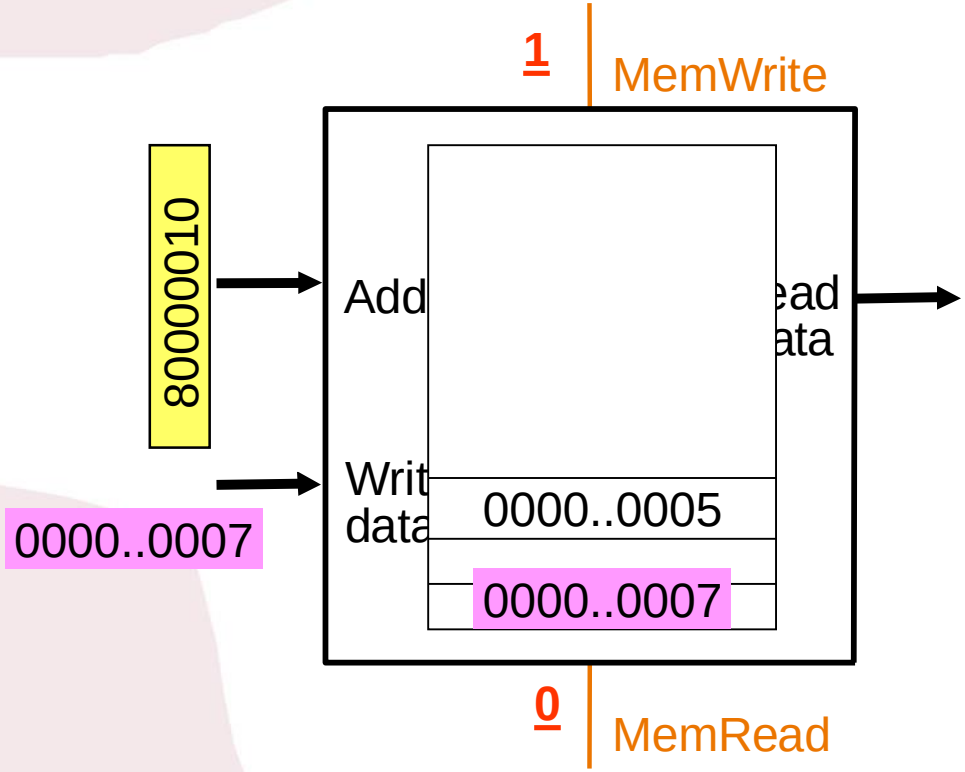


Componentes Básicos: Acesso à memória



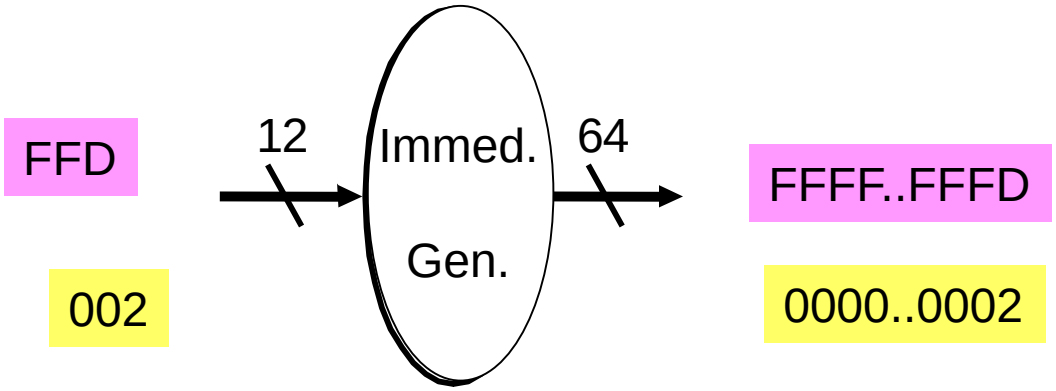
Memória de Dados - Leitura

Componentes Básicos: Acesso à memória



Memória de Dados - Escrita

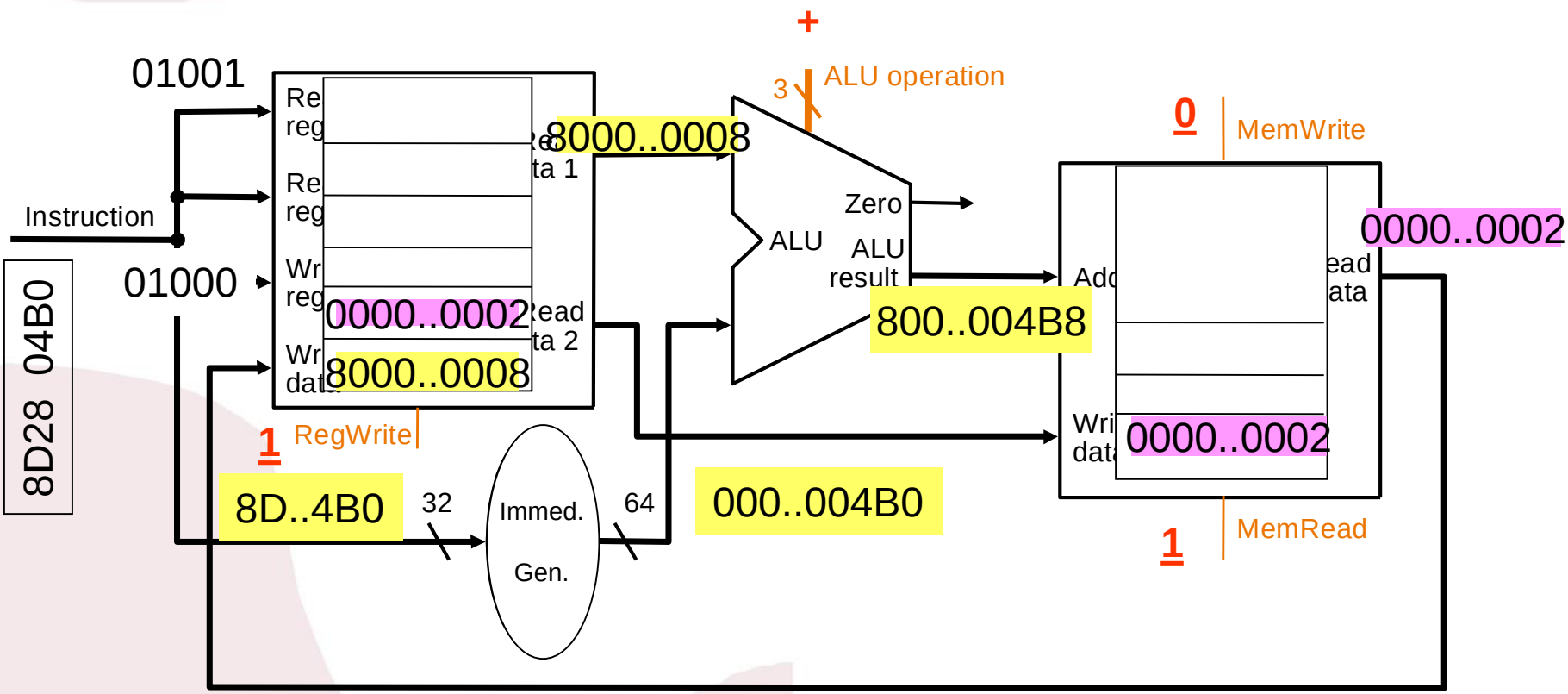
Componentes Básicos: Acesso à memória



Geração de Constantes: extrai e compõe constante da instrução preservando o sinal

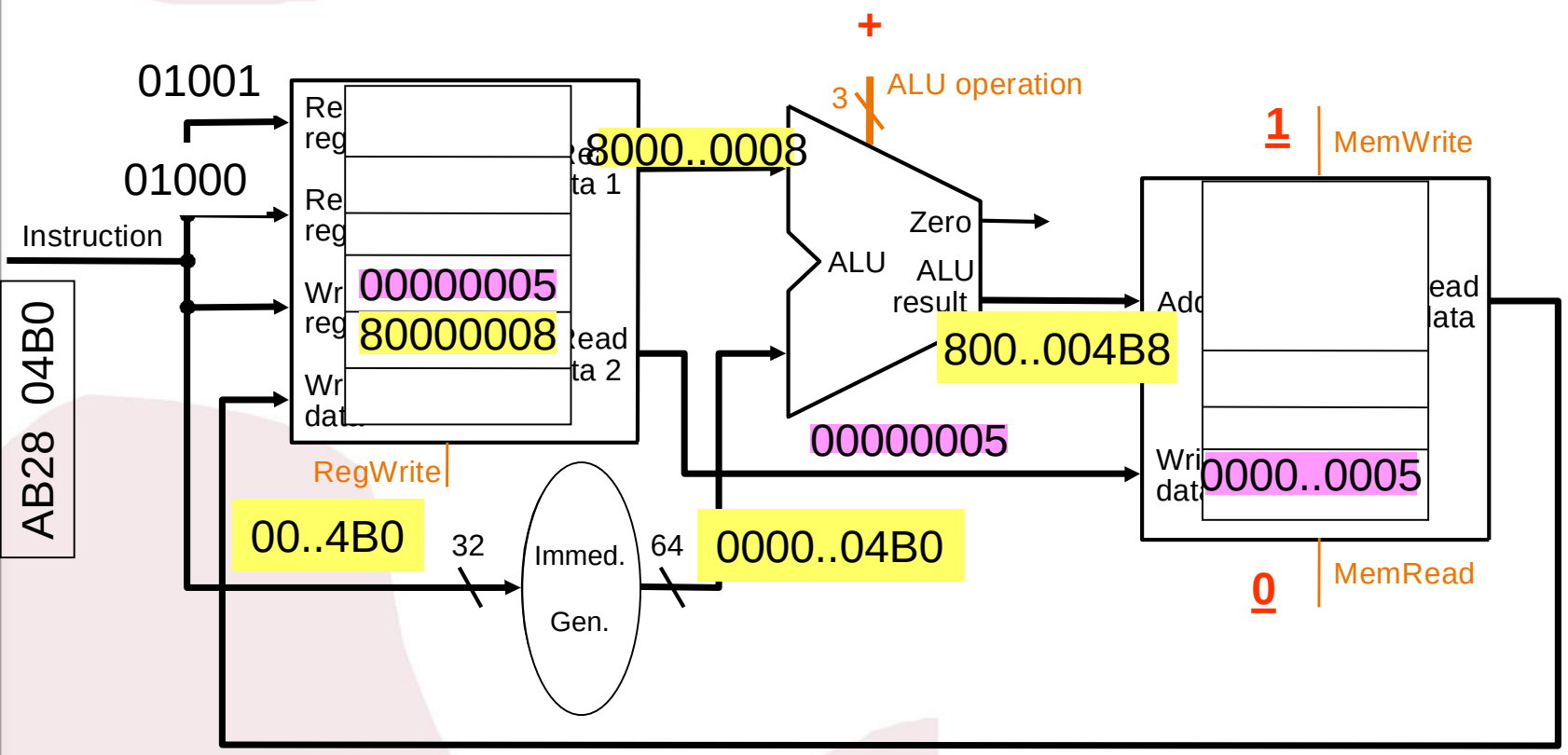
Instruções Load/Store

ld x8, desl(x9)



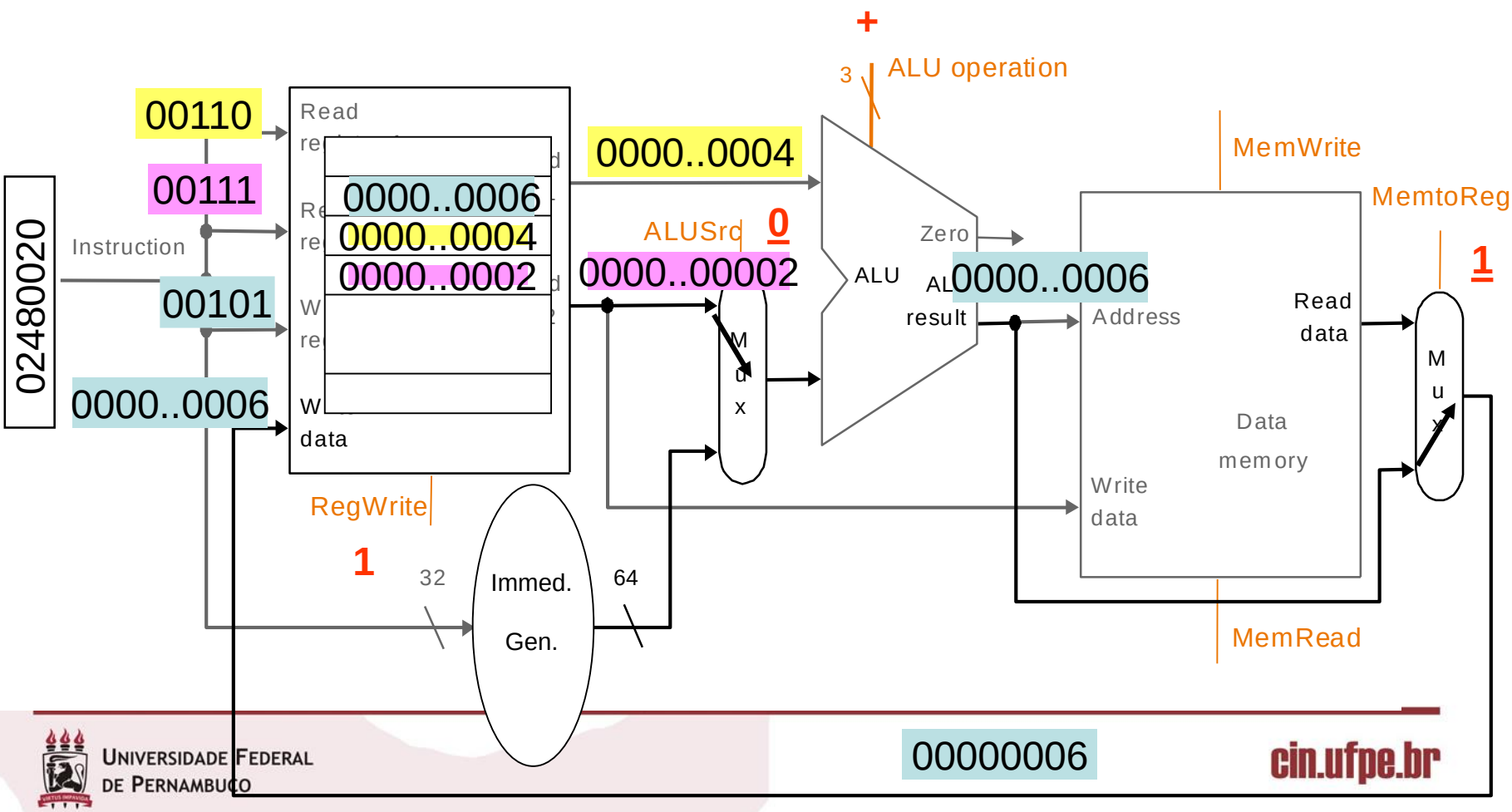
Instruções Load/Store

sw x8, desl(x9)



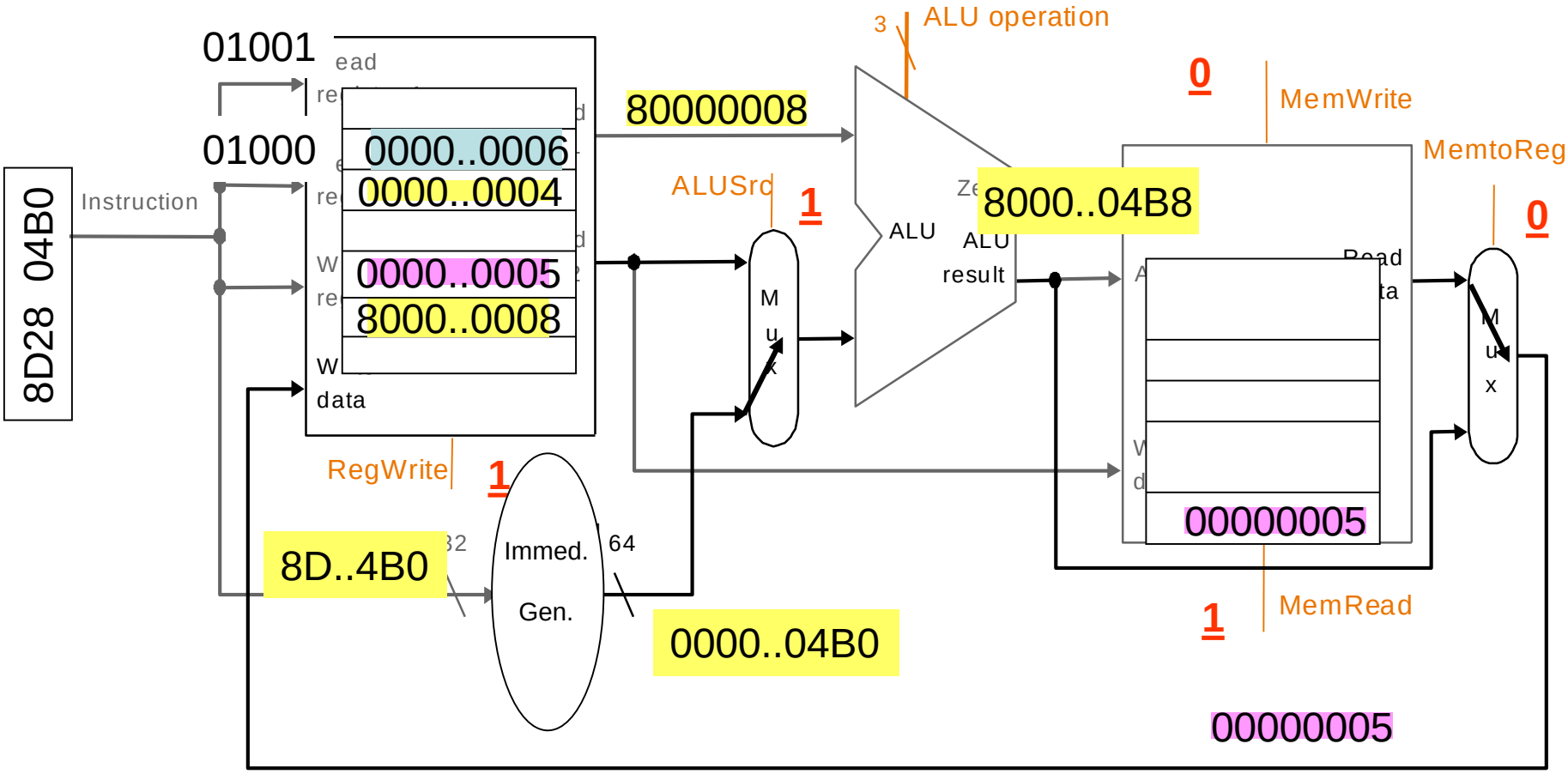
Instruções Aritméticas e de Load/Store

add x5, x6, x7



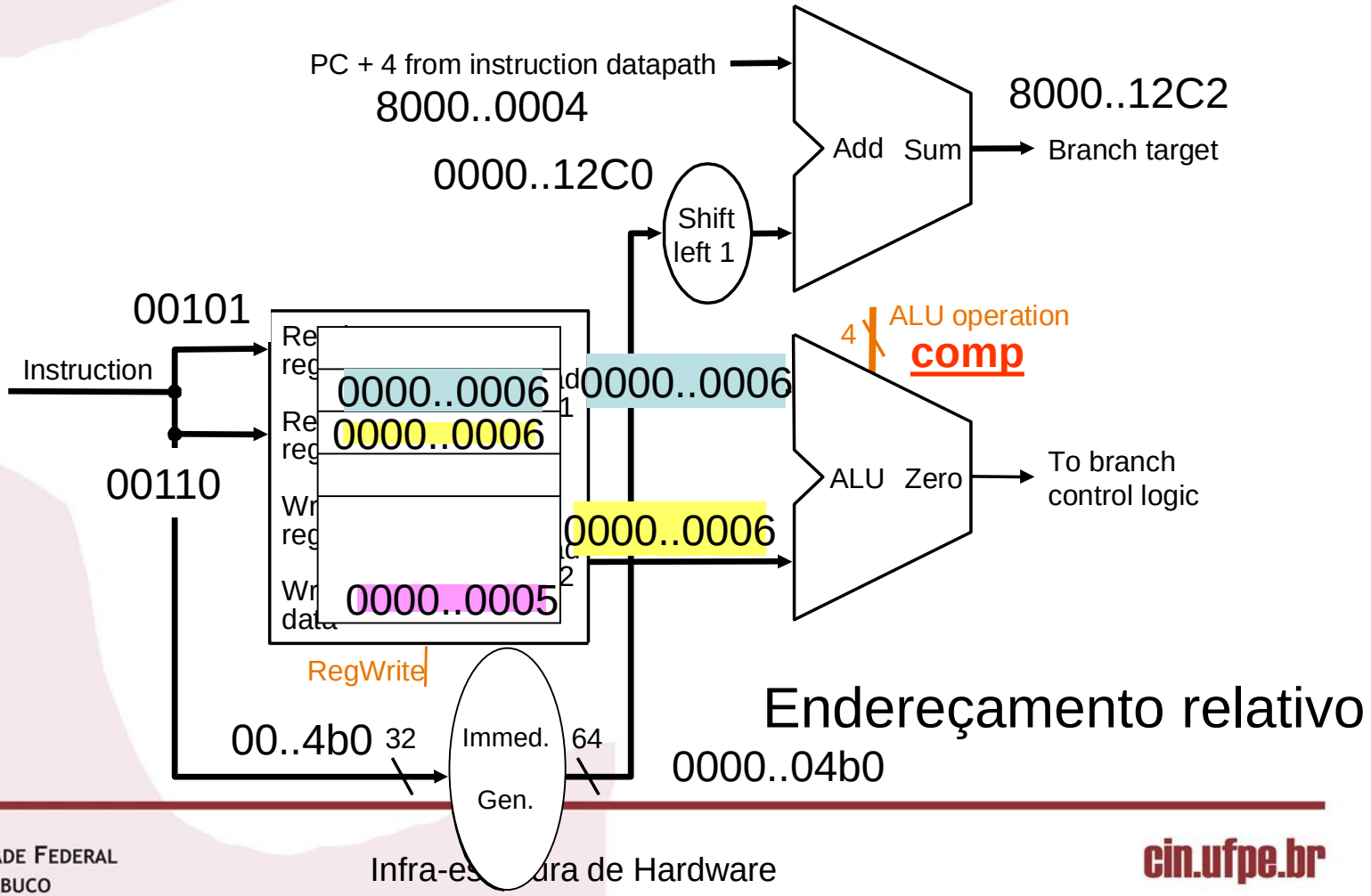
Instruções Aritméticas e de Load/Store

ld x8, desl(x9)



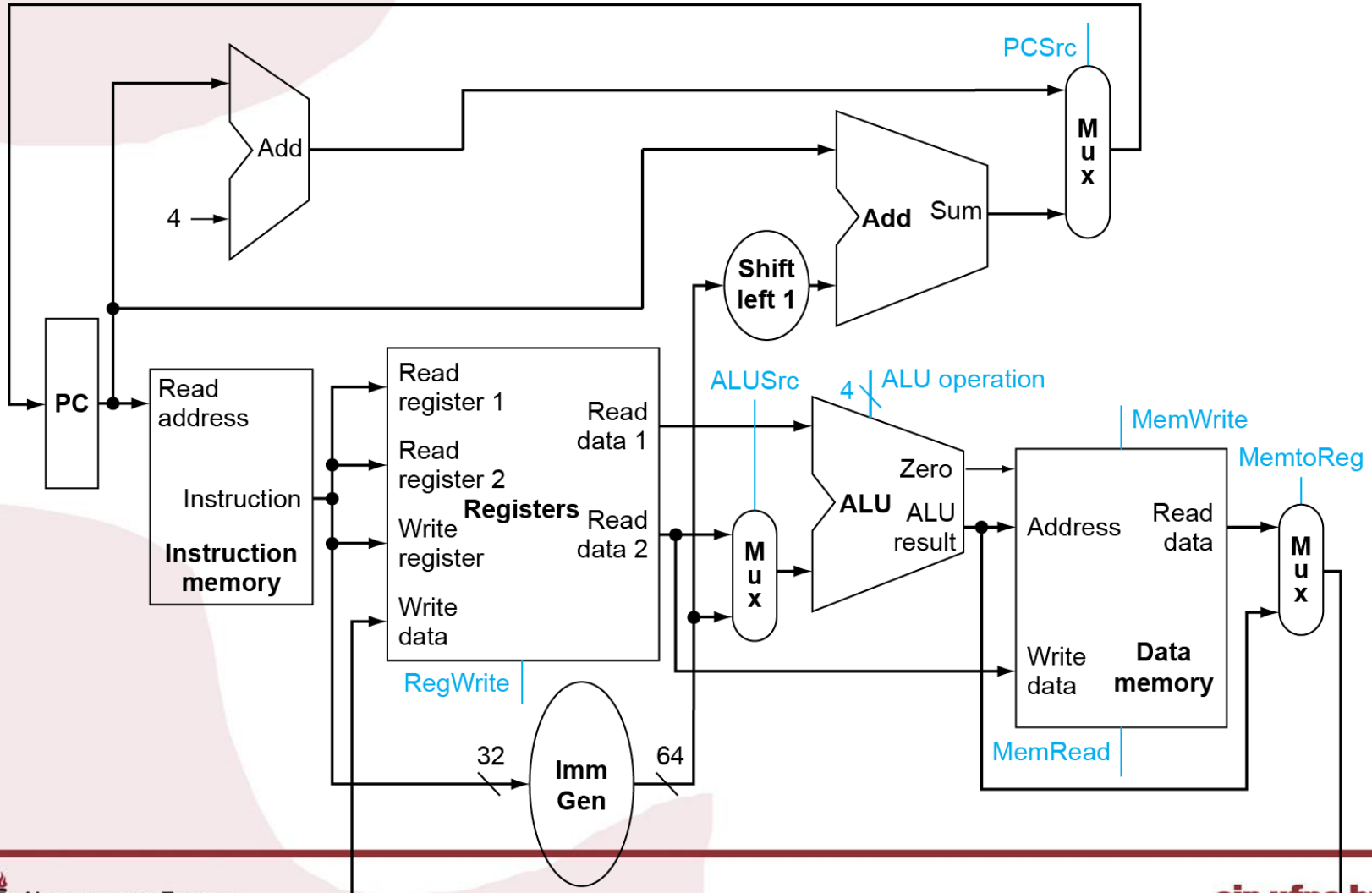
Instrução de Branch on equal

Beq x5,x6, end.



Endereçamento relativo

Unidade Processamento



Controle da ALU

- ALU usada para
  - Load/Store: F = adição
  - Branch: F = subtração
  - R-type: F depende do opcode

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract

Controle da ALU

- Suponha que o ALUOp de 2 bits seja derivado do opcode
  - Lógica combinacional deriva o controle da ALU

opcode	ALUOp	Operation	Opcode field	ALU function	ALU control
ld	00	load register	XXXXXXXXXXX	add	0010
sd	00	store register	XXXXXXXXXXX	add	0010
beq	01	branch on equal	XXXXXXXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001

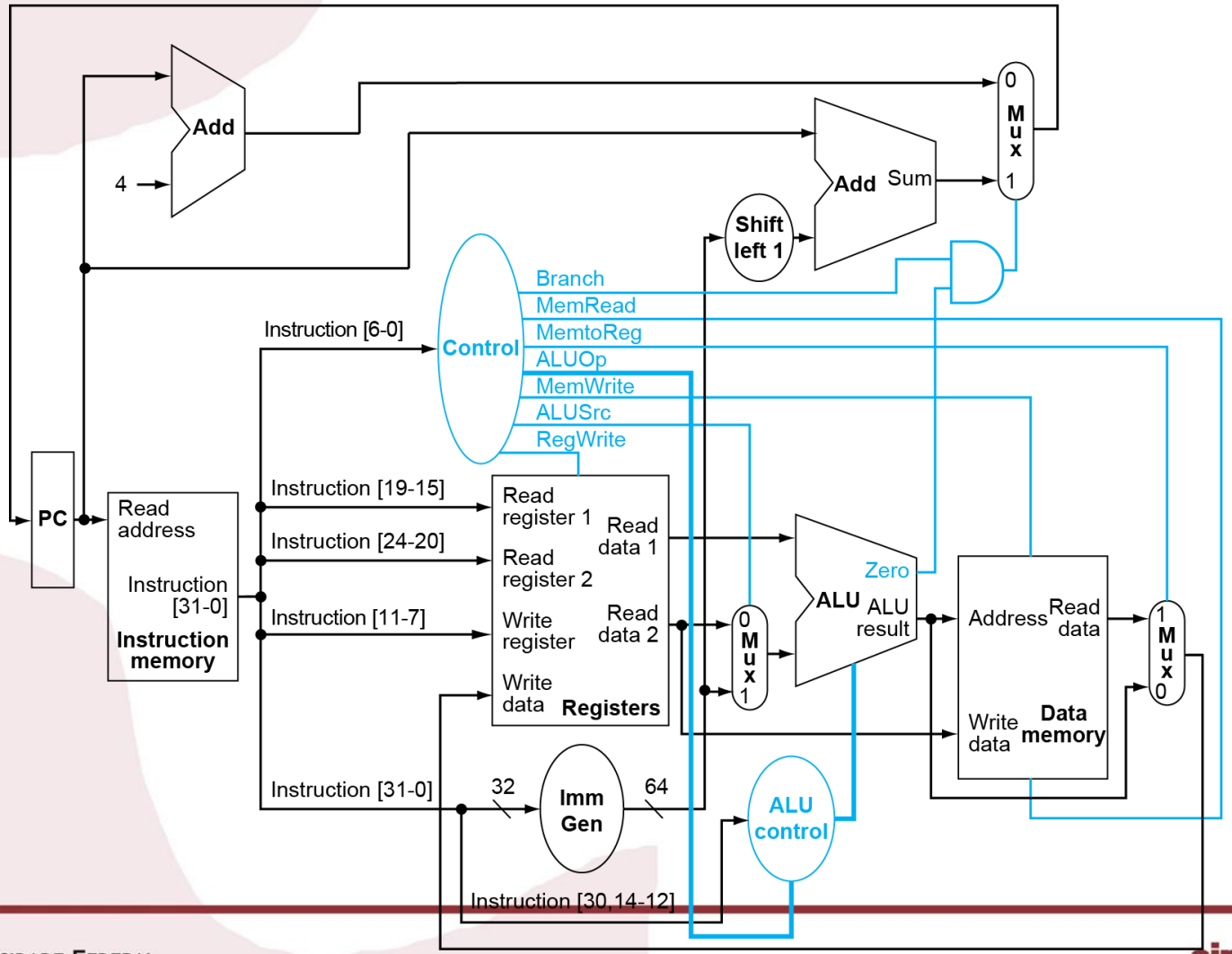
Unidade de Controle

- Sinais de controle derivados da instrução

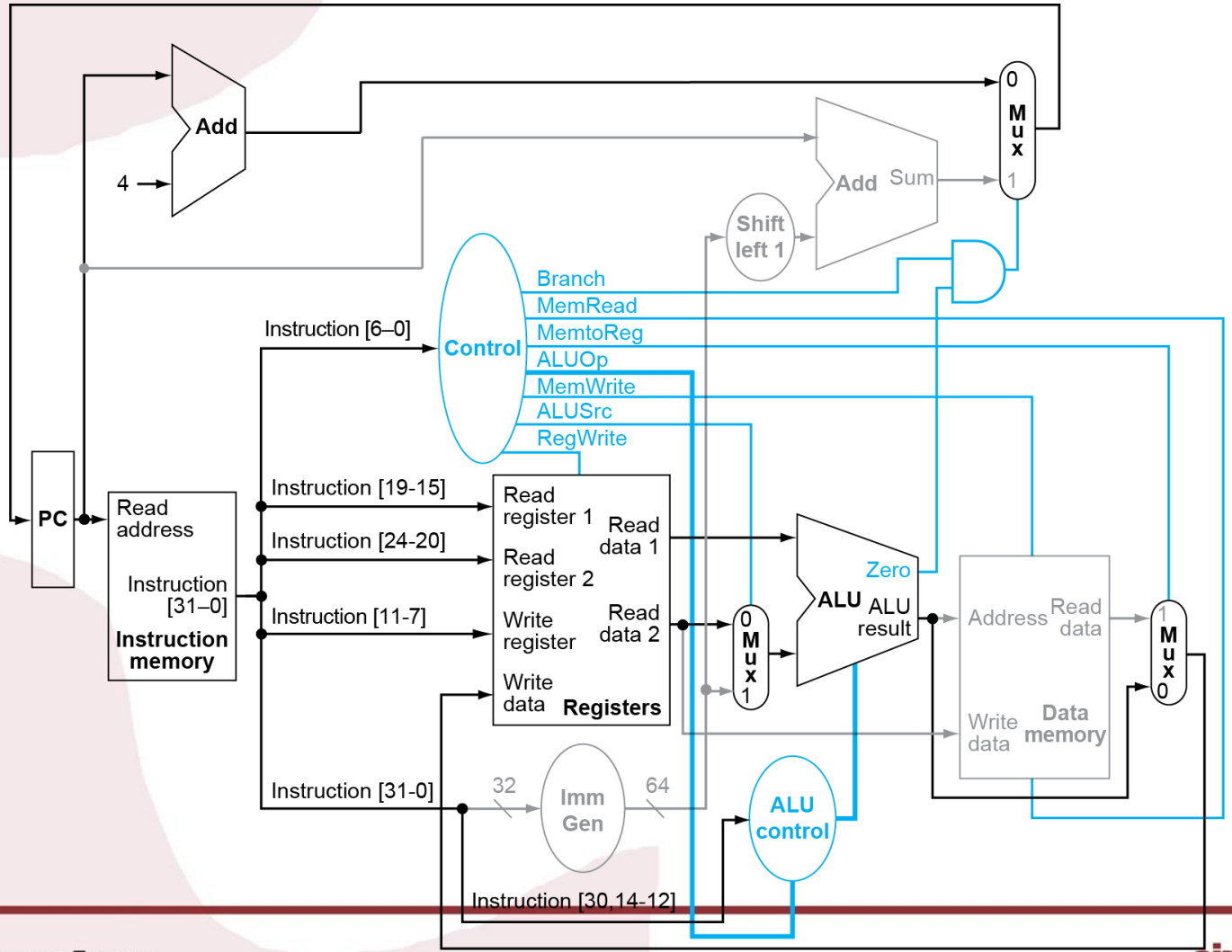
Name (Bit position)	Fields 31:25	24:20	19:15	14:12	11:7	6:0
(a) R-type	funct7	rs2	rs1	funct3	rd	opcode
(b) I-type	immediate[11:0]		rs1	funct3	rd	opcode
(c) S-type	immed[11:5]	rs2	rs1	funct3	immed[4:0]	opcode
(d) SB-type	immed[12,10:5]	rs2	rs1	funct3	immed[4:1,11]	opcode

ALUOp		Funct7 field							Funct3 field			
ALUOp1	ALUOp0	I[31]	I[30]	I[29]	I[28]	I[27]	I[26]	I[25]	I[14]	I[13]	I[12]	Operation
0	0	X	X	X	X	X	X	X	X	X	X	0010
X	1	X	X	X	X	X	X	X	X	X	X	0110
1	X	0	0	0	0	0	0	0	0	0	0	0010
1	X	0	1	0	0	0	0	0	0	0	0	0110
1	X	0	0	0	0	0	0	0	1	1	1	0000
1	X	0	0	0	0	0	0	0	1	1	0	0001

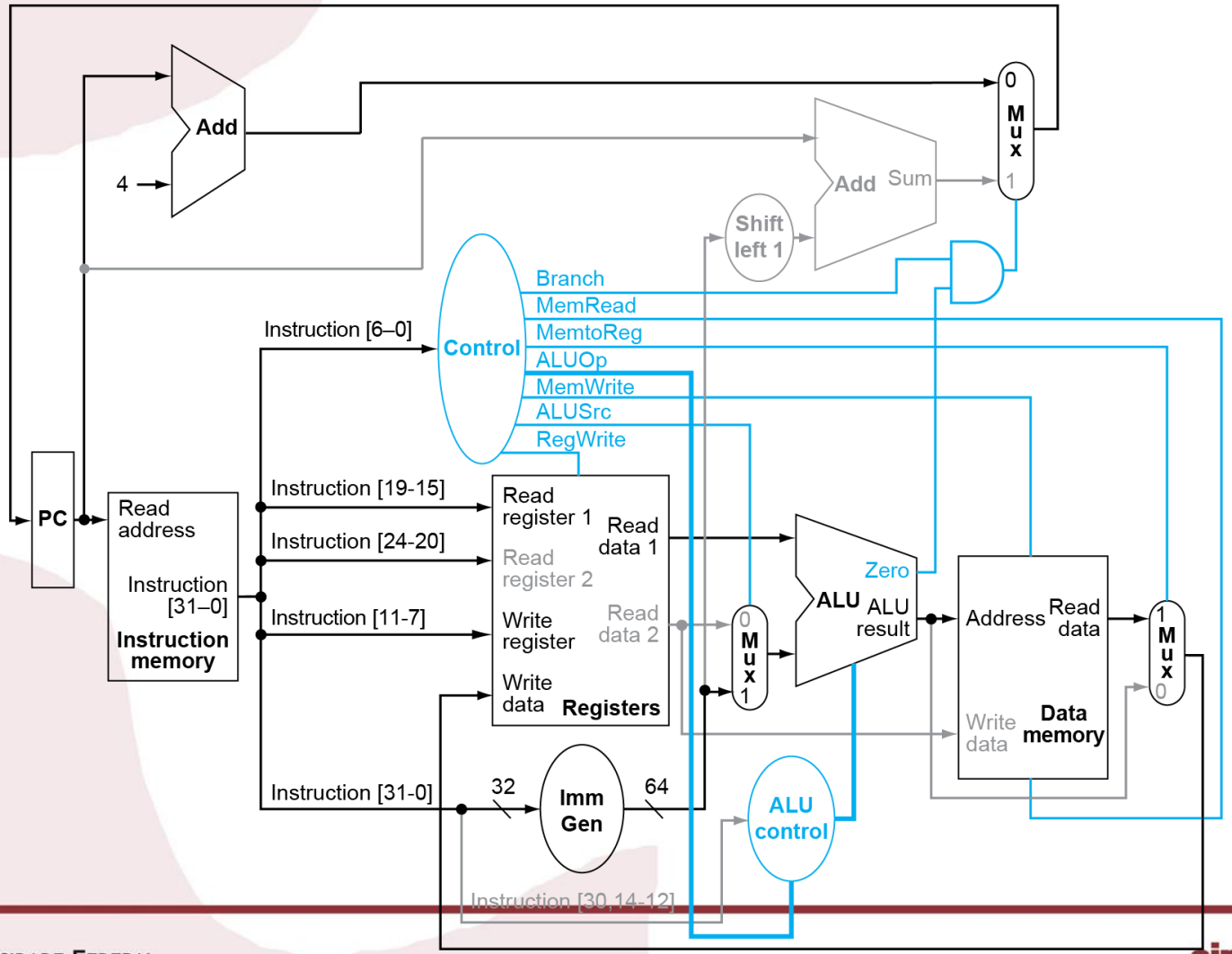
Processamento e Controle



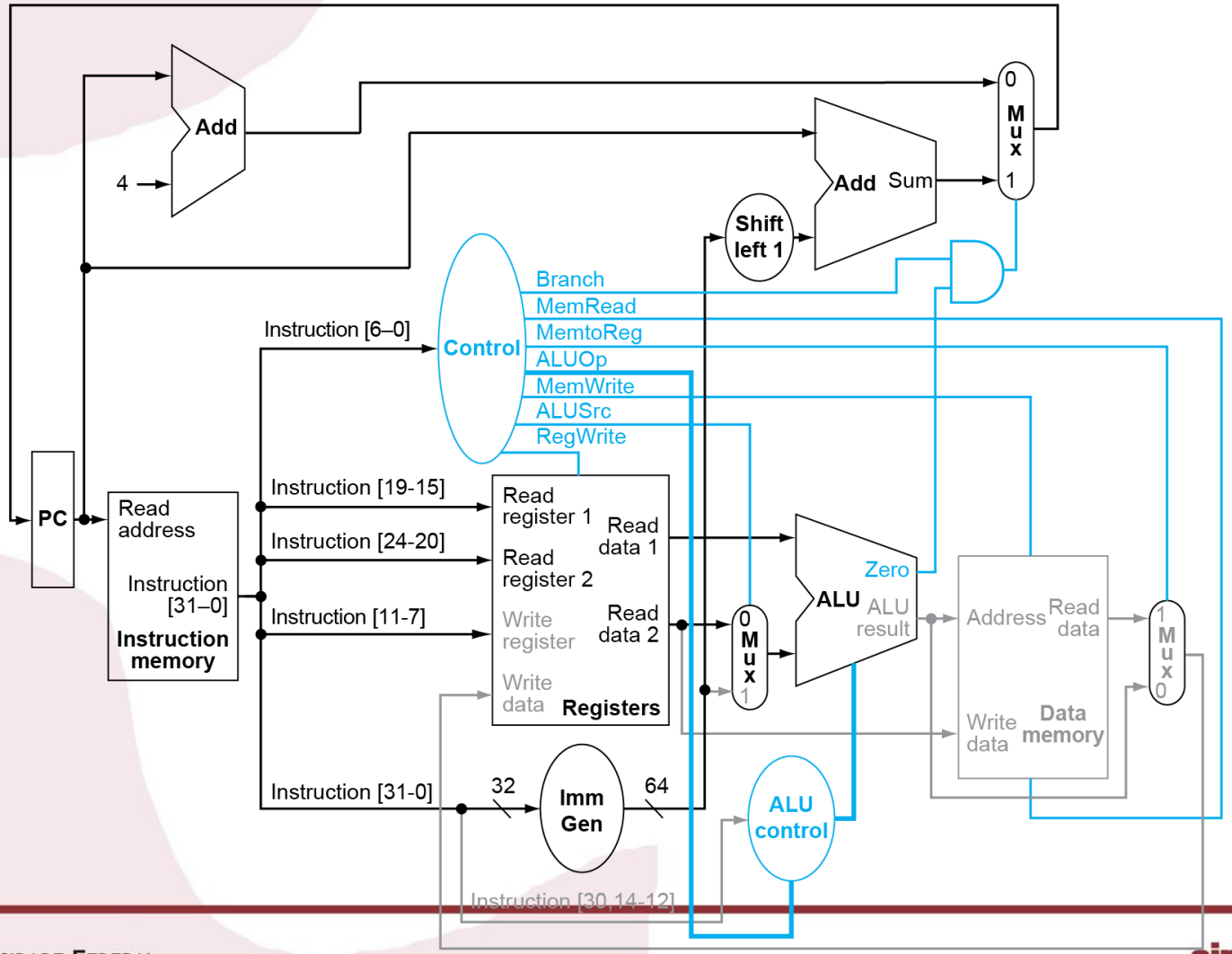
Instrução R-Type



Instrução Load



Instrução BEQ



Análise de Desempenho

- Mono-ciclo:
  - Período do relógio definido em função da duração da instrução mais lenta
    - 8ns (5ns)
    - Implementação pouco eficiente
    - CPU_{time} = nr. Instruções x período_clock
- Como melhorar o desempenho na execução de várias instruções?
  - Multi-ciclo: cada estágio é executado em um ciclo do relógio.
  - Começar uma instrução ANTES da última instrução iniciada terminar.